```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

D 10

```
**FILE**ID**RMOFABCHK
```

```
RRRRRRRR    MM      MM    000000    FFFFFFFFFF    AAAAAA    BBBBBBBB      CCCCCCCC  HH      HH  KK      KK
RRRRRRRR    MM      MM    000000    FFFFFFFFFF    AAAAAA    BBBBBBBB      CCCCCCCC  HH      HH  KK      KK
RR      RR  MMMM  MMMM  00      00  FF          AA      AA  BB      BB  CC          HH      HH  KK      KK
RR      RR  MMMM  MMMM  00      00  FF          AA      AA  BB      BB  CC          HH      HH  KK      KK
RR      RR  MM  MM  MM  00    0000  FF          AA      AA  BB      BB  CC          HH      HH  KK    KK
RR      RR  MM  MM  MM  00    0000  FF          AA      AA  BB      BB  CC          HH      HH  KK    KK
RRRRRRRR    MM      MM  00  00  00  FFFFFFFF    AA      AA  BBBBBBBB    CC          HHHHHHHHHH  KKKKKK
RRRRRRRR    MM      MM  00  00  00  FFFFFFFF    AA      AA  BBBBBBBB    CC          HHHHHHHHHH  KKKKKK
RR  RR      MM      MM  0000    00  FF          AAAAAAAAAA  BB      BB  CC          HH      HH  KK    KK
RR  RR      MM      MM  0000    00  FF          AAAAAAAAAA  BB      BB  CC          HH      HH  KK    KK
RR    RR    MM      MM  00      00  FF          AA      AA  BB      BB  CC          HH      HH  KK      KK      ....
RR    RR    MM      MM  00      00  FF          AA      AA  BB      BB  CC          HH      HH  KK      KK      ....
RR      RR  MM      MM    000000    FF          AA      AA  BBBBBBBB      CCCCCCCC  HH      HH  KK      KK      ....
RR      RR  MM      MM    000000    FF          AA      AA  BBBBBBBB      CCCCCCCC  HH      HH  KK      KK      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            $BEGIN  RMOFABCHK,000,RM$RMS0,<COMMON FAB CHECKING>
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000    ;
0000    ;++
0000    ; Facility: rms32
0000    ;
0000    ; Abstract:
0000    ;               this routine performs common fab call argument
0000    ;               list and fab validation.
0000    ;
0000    ; Environment:
0000    ;               star processor running starlet exec.
0000    ;
0000    ; Author: L F Laverdure,          creation date: 4-JAN-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-004 DGB0043         Donald G. Blair         02-May-1984
0000    ;               If the PIO$V_INHAST bit is set when we start an
0000    ;               RMS operation, we conclude that the caller must be
0000    ;               at exec AST level or higher and that he would break
0000    ;               RMS synchronization rules if allowed to continue.
0000    ;               Return Error.
0000    ;
0000    ;       V03-003 RAS0171         Ron Schaefer            19-Jul-1983
0000    ;               Change RAS0162 to a new specific structure-less
0000    ;               error.
0000    ;
0000    ;       V03-002 RAS0162         Ron Schaefer            17-Jun-1983
0000    ;               Detect and report the AST/non-AST caller's mode
0000    ;               wait hang condition, by checking the low bit of the
0000    ;               FAB's BLN field.
0000    ;
```

```
                    0000    ;         V03-001 KBT0206         Keith B. Thompson        23-Aug-1982
                    0000    ;                 Reorganize psects
                    0000    ;
                    0000    ;         V007    REFORMAT        Ken Henderson   29-JUL-1980     15:42
                    0000    ;                 code was reformatted
                    0000    ;
                    0000    ;--
                    0000
```

```
0000            .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; Include Files:
0000    ;
0000
0000    ;
0000    ; Macros:
0000    ;
0000
0000            $FABDEF
0000            $PIODEF
0000            $PSLDEF
0000            $RMSDEF
0000
0000    ;
0000    ; Equated Symbols:
0000    ;
0000
0000    ;
0000    ; Own Storage:
0000    ;
0000
```

```
0000            .SBTTL  RM$FABCHK - COMMON ARGLIST AND FAB VALIDATION ROUTINE
0000
0000     ;++
0000     ; RM$FABCHK
0000     ;
0000     ;  this routine performs the following functions:
0000     ;
0000     ;       1. setup r11 to point to the image i/o impure area
0000     ;          (it may be changed in fset to the process i/o impure area)
0000     ;       2. check argument list for accessibility and validity
0000     ;       3. check fab for accessibility and validity
0000     ;       4. set r8 to address of fab
0000     ;       5. clear sts and stv
0000     ;       6. inhibit rms internal asts
0000     ;       7. set r9 to the value of ifi
0000     ;
0000     ;
0000     ;
0000     ; Calling sequence:
0000     ;
0000     ;       bsbw    rm$fabchk
0000     ;
0000     ; Input Parameters:
0000     ;
0000     ;       ap      arglist addr
0000     ;
0000     ; Implicit Inputs:
0000     ;
0000     ;       the contents of the arglist and the bid, bln, and ifi fields
0000     ;       of the fab.
0000     ;
0000     ; Output Parameters:
0000     ;
0000     ;       r11     impure area address
0000     ;       r9      ifi value
0000     ;       r8      fab address
0000     ;       r7      mode of caller
0000     ;
0000     ; Implicit Outputs:
0000     ;
0000     ;       the sts and stv fields of the fab are zeroed.
0000     ;
0000     ; Completion Codes:
0000     ;
0000     ;       z-bit set if ifi = zero, else clear.
0000     ;       if any errors, the rms error code is set into r0
0000     ;       and return is made to the user (not caller).
0000     ;
0000     ; Side Effects:
0000     ;
0000     ;       rms internal asts are inhibited.
0000     ;
0000     ;--
0000
```

```
                       0000
                       0000  ;
                       0000  ;  set up pointer to impure area based on the mode of the caller
                       0000  ;
                       0000
                       0000  RM$FABCHK::
                5B  DC 0000          MOVPSL  R11                          ; get psl
  57   5B   02  16  EF 0002          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R11,R7
                       0007                                               ; extract the previous mode
  5B   00000000'9F  DE 0007          MOVAL   @#PIO$GW_IIOIMPA,R11         ; image io impure area addr
                       000E
                       000E  ;
                       000E  ;  perform accessibility checks
                       000E  ;
                       000E
        58   04 AC  D0 000E          MOVL    4(AP),R8                     ; get fab address
                       0012          IFNOWRT #FAB$C_BLN,(R8),ERRFAB       ; fab writeable?
                       001A          ASSUME  FAB$B_BID EQ 0
            03   68  91 001A          CMPB    (R8),#FAB$C_BID              ; is it a fab?
                 29  12 001D          BNEQ    ERRFAB
     50 8F   01 A8  91 001F          CMPB    FAB$B_BLN(R8),#FAB$C_BLN; is it long enough?
                 1B  1F 0024          BLSSU   ERRBLN
         10 01 A8  E8 0026          BLBS    FAB$B_BLN(R8),ERRACT         ; is this FAB busy?
                       002A                                               ; continue if not
                       002A
                       002A  ;
                       002A  ;  zero the sts and stv fab fields
                       002A  ;
                       002A
                       002A          ASSUME  FAB$L_STS+4 EQ FAB$L_STV
            08 A8  7C 002A          CLRQ    FAB$L_STS(R8)
                       002D
                       002D  ;
                       002D  ;  Disable AST's.  If the PIO$V_INHAST bit is already set, we
                       002D  ;  conclude that the caller must be at exec ast level or higher
                       002D  ;  (otherwise, he could not have kicked off an RMS operation
                       002D  ;  while RMS was already in progress) and would break RMS
                       002D  ;  synchronization rules if allowed to continue.  Return RMS$_BUSY
                       002D  ;  status when this happens.
                       002D  ;
                       002D
                00  E2 002D          BBSS    #PIO$V_INHAST,-              ; set inhast bit.  err if already set.
     05 00000000'9F     002F                  @#PIO$GW_STATUS,ERRACT
        59   02 A8  3C 0035          MOVZWL  FAB$W_IFI(R8),R9             ; set r9 = ifi value
                    05 0039          RSB
                       003A
                       003A
                       003A  ;
                       003A  ;  an error has occurred in validating the argument list or fab
                       003A  ;
                       003A  ;  since an error code cannot be safely stored in the fab,
                       003A  ;  no attempt to generate an err= ast will be made.
                       003A  ;  r0 will be set to the appropriate error code and an
                       003A  ;  exception, if enabled, will be generated upon ret.
                       003A  ;
                       003A
```

```
                  003A      ERRACT:
                  003A                 RMSERR  BUSY                           ; Syncrhonization problem
     0C     11    003F                 BRB     BASIC_ERR
                  0041
                  0041      ERRBLN:
                  0041                 RMSERR  BLN                            ; invalid block length
     05     11    0046                 BRB     BASIC_ERR
                  0048
                  0048      ERRFAB:
                  0048                 RMSERR  FAB                            ; invalid fab
                  004D
                  004D      BASIC_ERR:
                  004D                 SSB     #16,R0                         ; prefix the facility code
                  0051                                                        ; ... to the error code
            04    0051                 RET                                    ; and return to caller
                  0052
                  0052                 .END
```

```
RM
Sy

$$
$$
$$
$$
$$
$$
$$
FA
FC
FC
FI
FW
IF
IF
IF
IF
IF
IO
IR
IR
IR
MA
PI
PU
QI
QI
QI
QI
QI
QI
QI
QI
QI
QI
QI
QI
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
RM
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BASIC_ERR                         0000004D R      01
ERRACT                            0000003A R      01
ERRBLN                            00000041 R      01
ERRFAB                            00000048 R      01
FAB$B_BID                       = 00000000
FAB$B_BLN                       = 00000001
FAB$C_BID                       = 00000003
FAB$C_BLN                       = 00000050
FAB$L_STS                       = 00000008
FAB$L_STV                       = 0000000C
FAB$W_IFI                       = 00000002
PIO$GW_IIOIMPA                    ********    X   01
PIO$GW_STATUS                     ********    X   01
PIO$V_INHAST                    = 00000000
PSL$S_PRVMOD                    = 00000002
PSL$V_PRVMOD                    = 00000016
RM$FABCHK                         00000000 RG     01
RMS$_BLN                        = 0001842C
RMS$_BUSY                       = 0001848C
RMS$_FAB                        = 0001850C

                                                +------------------+
                                                ! Psect synopsis !
                                                +------------------+

PSECT name                      Allocation           PSECT No.  Attributes
----------                      ----------           ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RM$RMS0                         00000052  (    82.)  01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                             +------------------------+
                                             ! Performance indicators !
                                             +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      29    00:00:00.10    00:00:01.21
Command processing                 115    00:00:00.74    00:00:06.48
Pass 1                             206    00:00:04.66    00:00:15.03
Symbol table sort                    0    00:00:00.39    00:00:00.64
Pass 2                              52    00:00:01.01    00:00:02.53
Symbol table output                  5    00:00:00.05    00:00:00.37
Psect synopsis output                1    00:00:00.02    00:00:00.02
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               410    00:00:06.97    00:00:26.28

The working set limit was 1350 pages.
24174 bytes (48 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 453 non-local and 1 local symbols.
219 source lines were read in Pass 1, producing 13 object records in Pass 2.
18 pages of virtual memory were used to define 17 macros.
```

```
                                          +----------------------------+
                                          ! Macro library statistics !
                                          +----------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                          7
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       5
TOTALS (all libraries)                                 13

558 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOFABCHK/OBJ=OBJ$:RMOFABCHK MSRC$:RMOFABCHK/UPDATE=(ENH$:RMOFABCHK)+EXECML$/LIB+LIB$:RMS/LIB
```

0318 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM0EXTRMS
LIS

RM0DIRSCN
LIS

RM0CRECOM
LIS

RM0FABCHK
LIS

RM0CHKSUM
LIS

RM0FWASET
LIS

RM0EXTEND
LIS

RM0IFISI
LIS

RM0JOURNL
LIS

RM0FSETI
LIS

RM0FSET
LIS

RM0COMCLN
LIS

RM0DUMMY
LIS

RM0FILFNC
LIS